➢ Verilog Code of CPU:

```
// Program ----> Obtaining the 2's complement of any number then close.                    //
////////////////////////////////////////////////////////CPU////////////////////////////////////////////////////////
//*****************************************************High Level Module*****************************************************//
module CPU (clk,clr,
        AC_Load
        );

  //Input Wires.....................................................
  input clk,clr;
  //Output Registars.....................................................
  output [15:0] AC_Load;
  //Data_output type.....................................................
  reg T0,T1,T2,T3,T4,T5,T6,T7;
  reg D0,D1,D2,D3,D4,D5,D6,D7;
  reg [15:0] DataIn_Instruction;
  reg [2:0] counterOut;
  reg [11:0] PC_Loading;
  reg [11:0] AR_Address_IR;
  wire [15:0] Instruction;
  reg [11:0] Instruction_Address;
  wire [15:0] Data;
  reg [15:0] DataInput;
  reg [11:0] Data_Address;
  reg read,write;
  reg [15:0] IR_Stored;
  reg [11:0] DraftAddress;
  reg [11:0] DraftAddress2;
  reg E;
  reg I;
  integer i;
  reg [15:0] AC_Load;
  reg [16:0] AC_Result;
  reg [15:0] DR_Store;
```

```
  reg [15:0] MdataIn;
  reg G;
  wire r;
//
assign r= D7&T3&~I;
//
Data_Memory D (clk,read,write,DataInput,Data_Address,Data);
Instruction_Memory J (clk,Instruction_Address,Instruction);
////////////////////////////////////////////////Sequential Counter////////////////////////////////////////////////
initial
begin
E=0;
read=1;
write=1;
counterOut =3'b000;
PC_Loading=12'h000; //Address = 000 H.....
AC_Load=17'b00000000000000000;
end
always @ (posedge clk) begin
if(clr==1 )
counterOut=3'b000;
else
counterOut= counterOut+3'b001;
end
////////////////////////////////////////////////Timing Signals Generation////////////////////////////////////////////////
                                    // Decoder 1 //
always @(*)
begin
case(counterOut)
4'b000:
begin
T0=1;T1=0;T2=0;T3=0;T4=0;T5=0;T6=0;T7=0;
end
4'b001:
begin
```

```
T0=0;T1=1;T2=0;T3=0;T4=0;T5=0;T6=0;T7=0;
end
4'b010:
begin
T0=0;T1=0;T2=1;T3=0;T4=0;T5=0;T6=0;T7=0;
end
4'b011:
begin
T0=0;T1=0;T2=0;T3=1;T4=0;T5=0;T6=0;T7=0;
end
4'b100:
begin
T0=0;T1=0;T2=0;T3=0;T4=1;T5=0;T6=0;T7=0;
end
4'b101:
begin
T0=0;T1=0;T2=0;T3=0;T4=0;T5=1;T6=0;T7=0;
end
4'b110:
begin
T0=0;T1=0;T2=0;T3=0;T4=0;T5=0;T6=1;T7=0;
end
default:
begin
T0=0;T1=0;T2=0;T3=0;T4=0;T5=0;T6=0;T7=1;
end
endcase
end
//////////////////////////////////////////////////Decoding Instruction Signal//////////////////////////////////////////
                                        // Decoder 2 //
always @(*) begin
if(DataIn_Instruction[12]==0 && DataIn_Instruction[13]==0 && DataIn_Instruction[14]==0)
begin
D0=1;D1=0;D2=0;D3=0;D4=0;D5=0;D6=0;D7=0;
end
```

```
else if(DataIn_Instruction[12]==1 && DataIn_Instruction[13]==0 && DataIn_Instruction[14]==0)
begin
D0=0;D1=1;D2=0;D3=0;D4=0;D5=0;D6=0;D7=0;
end
else if(DataIn_Instruction[12]==0 && DataIn_Instruction[13]==1 && DataIn_Instruction[14]==0)
begin
D0=0;D1=0;D2=1;D3=0;D4=0;D5=0;D6=0;D7=0;
end
else if(DataIn_Instruction[12]==1 && DataIn_Instruction[13]==1 && DataIn_Instruction[14]==0)
begin
D0=0;D1=0;D2=0;D3=1;D4=0;D5=0;D6=0;D7=0;
end
else if(DataIn_Instruction[12]==0 && DataIn_Instruction[13]==0 && DataIn_Instruction[14]==1)
begin
D0=0;D1=0;D2=0;D3=0;D4=1;D5=0;D6=0;D7=0;
end
else if(DataIn_Instruction[12]==1 && DataIn_Instruction[13]==0 && DataIn_Instruction[14]==1)
begin
D0=0;D1=0;D2=0;D3=0;D4=0;D5=1;D6=0;D7=0;
end
else if(DataIn_Instruction[12]==0 && DataIn_Instruction[13]==1 && DataIn_Instruction[14]==1)
begin
D0=0;D1=0;D2=0;D3=0;D4=0;D5=0;D6=1;D7=0;
end
else
begin
D0=0;D1=0;D2=0;D3=0;D4=0;D5=0;D6=0;D7=1;
end
end
//////////////////////////////////////////////////////Fetch//////////////////////////////////////////////////////////////
always @(posedge clk) begin
if(clr==1)
PC_Loading =12'h000;
if(T0==1)  AR_Address_IR = PC_Loading;
if (T1==1) begin
```

```
Instruction_Address = AR_Address_IR;
IR_Stored = Instruction;
DataIn_Instruction = IR_Stored;
PC_Loading = PC_Loading + 12'h001;
end
//////////////////////////////////////////////////////Decode//////////////////////////////////////////////////////
if(T2==1) begin
for(i=0;i<12;i=i+1) begin
Data_Address[i] = DataIn_Instruction[i];
end
I = DataIn_Instruction[15];
end
if((T3&I)==1) begin
for(i=0;i<12;i=i+1) begin
Data_Address[i] = Data[i];
end
end
//////////////////////////////////////////////////MRI Excute//////////////////////////////////////////////////////
//AND.....................
if((T4&D0)==1) DR_Store=Data;
if ((T5&D0)==1) begin
AC_Load=AC_Load&DR_Store;
counterOut=3'b000;
end
//ADD.....................
if ((T4&D1)==1) DR_Store=Data;
if ((T5&D1)==1) begin
AC_Result=AC_Load+DR_Store;
E=AC_Result[16];
AC_Load=AC_Result;
counterOut=3'b000;
end
//LDA.....................
if ((T4&D2)==1) DR_Store=Data;
if ((T5&D2)==1) begin
```

```
AC_Load=DR_Store;
counterOut=3'b000;
end
//STA.....
if ((T4&D3)==1) begin
DataInput=AC_Load;
counterOut=3'b000;
end
//BUN.....
if ((T4&D4)==1) begin
PC_Loading=Data_Address;
counterOut=3'b000;
end
//BSA.....
if ((T4&D5)==1) begin
DataInput=PC_Loading;
Data_Address=Data_Address+12'b000000000001;
end
if ((T5&D5)==1) begin
PC_Loading=Data_Address;
counterOut=3'b000;
end
//ISZ.....
if ((T4&D6)==1) begin
DR_Store=Data;
end
if ((T5&D6)==1) begin
DR_Store=DR_Store+16'b0000000000000001;
end
if ((T6&D6)==1) begin
DataInput=DR_Store;
if(DR_Store==16'b0000000000000000) PC_Loading=PC_Loading+12'b000000000001;
counterOut=3'b000;
end
//////////////////////////////////////////////////RRI Excute//////////////////////////////////////////////////////////
```

```
//CLA.....
if (r==1 && Data_Address==12'b100000000000 ) begin
AC_Load = 16'b0000000000000000;
end
//CLE.....
if (r==1 && Data_Address==12'b010000000000 ) begin
E=0;
end
//CMA.....
if (r==1 && Data_Address==12'b001000000000 ) begin
AC_Result=~AC_Load;
E=AC_Result[16];
AC_Load=AC_Result;
end
//CME.....
if (r==1 && Data_Address==12'b000100000000 ) begin
E=~E;
end
//CIR.....
if (r==1 && Data_Address==12'b000010000000 ) begin
G=AC_Load[0];
AC_Result= AC_Result>>1;
AC_Result[16]=G;
E=AC_Result[16];
AC_Load=AC_Result;
end
//CIL.....
if (r==1 && Data_Address==12'b000001000000 ) begin
G=AC_Load[16];
AC_Result= AC_Result>>1;
AC_Result[0]=G;
E=AC_Result[16];
AC_Load=AC_Result;
end
//INC.....
```

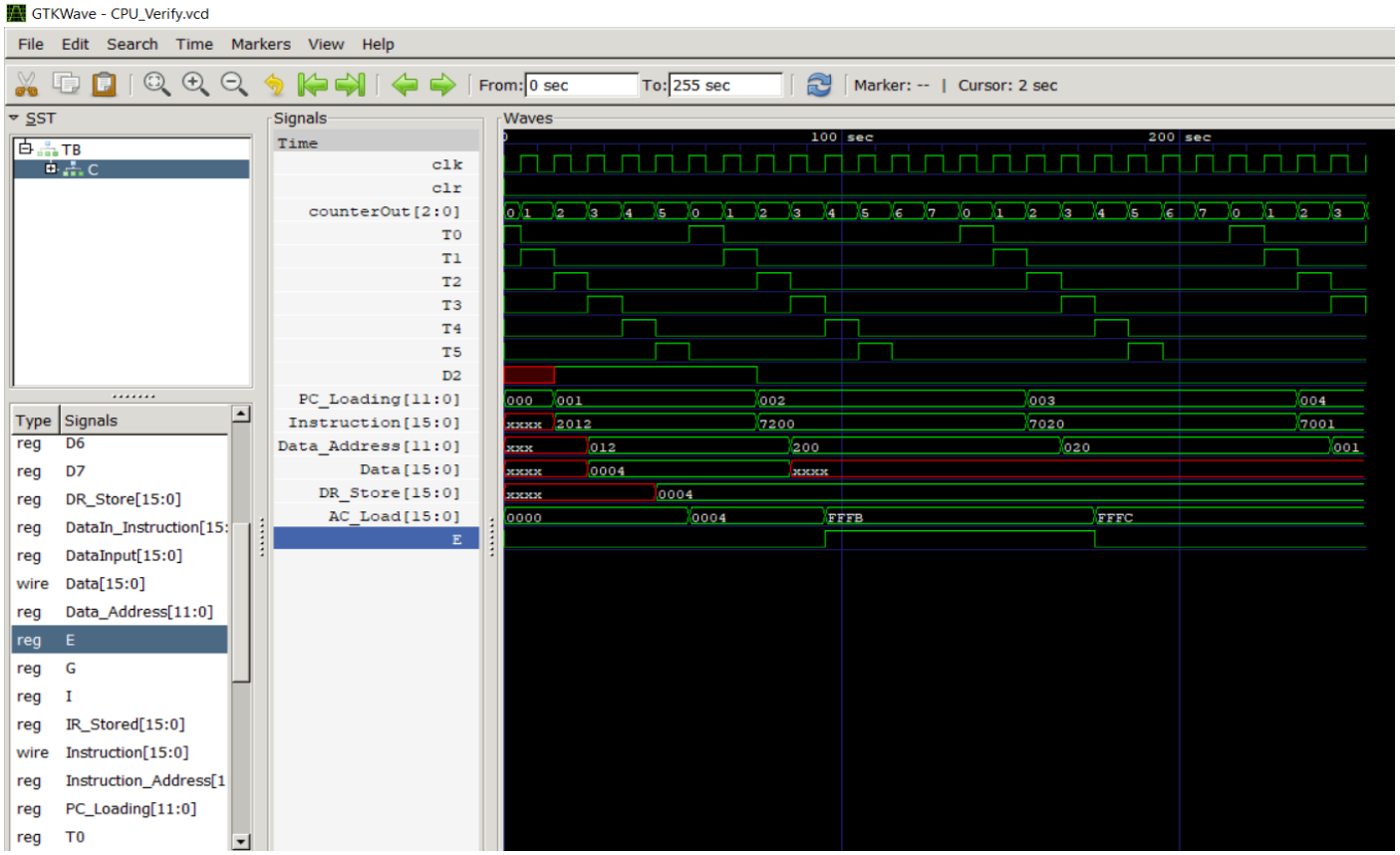
```
if (r==1 && Data_Address==12'b000000100000 ) begin
AC_Result=AC_Load+16'b0000000000000001;
E=AC_Result[16];
AC_Load=AC_Result;
end
//SPA.....
if (r==1 && Data_Address==12'b000000010000 ) begin
if(AC_Load[15]==0) PC_Loading = PC_Loading + 12'b000000000001;
end
//SNA.....
if (r==1 && Data_Address==12'b000000001000 ) begin
if(AC_Load[15]==1) PC_Loading = PC_Loading + 12'b000000000001;
end
//SZA.....
if (r==1 && Data_Address==12'b000000000100 ) begin
if(AC_Load==16'b0000000000000000) PC_Loading = PC_Loading + 12'b000000000001;
end
//SZE.....
if (r==1 && Data_Address==12'b000000000010 ) begin
if(E==0) PC_Loading = PC_Loading + 12'b000000000001;
end
//HLT.....
if (r==1 && Data_Address==12'b000000000001 ) begin
counterOut=3'b000;
$finish;
end
end
endmodule
//////////////////////////////////////////////////Memory//////////////////////////////////////////////////////////////
module Instruction_Memory(clk,PC_Address,instruction);
//Input ports.....
input clk;
input [11:0] PC_Address;
//Output ports.....
output [15:0] instruction;
```

```
reg [15:0] memory [0:4095];
integer i;
initial begin
//Instruction Initialized.....
memory[0]=16'h2012;
memory[1]=16'h7200;
memory[2]=16'h7020;
memory[3]=16'h7001;
for(i=0;i<4;i=i+1) begin
$display("memory[%0d]=0x%0h",i,memory[i]);
end
end
assign instruction =  memory[PC_Address];
endmodule
//.....//
module Data_Memory(clk,read,write,DataInput,Data_Address,Data);
//Input ports.....
input clk,read,write;
input [11:0] Data_Address;
input [15:0] DataInput;
//Output ports.....
output [15:0] Data;
reg [15:0] memory_Data [0:4095];
reg [15:0] Data;
integer i;
initial begin
for(i=0;i<1;i=i+1)
$display("memory_Data[%0d]=0x%0h",i,memory_Data[18]);
end
always @(posedge clk) begin
if (write)
memory_Data[Data_Address] <= DataInput;
if(read)
Data<=memory_Data[Data_Address];
end
```

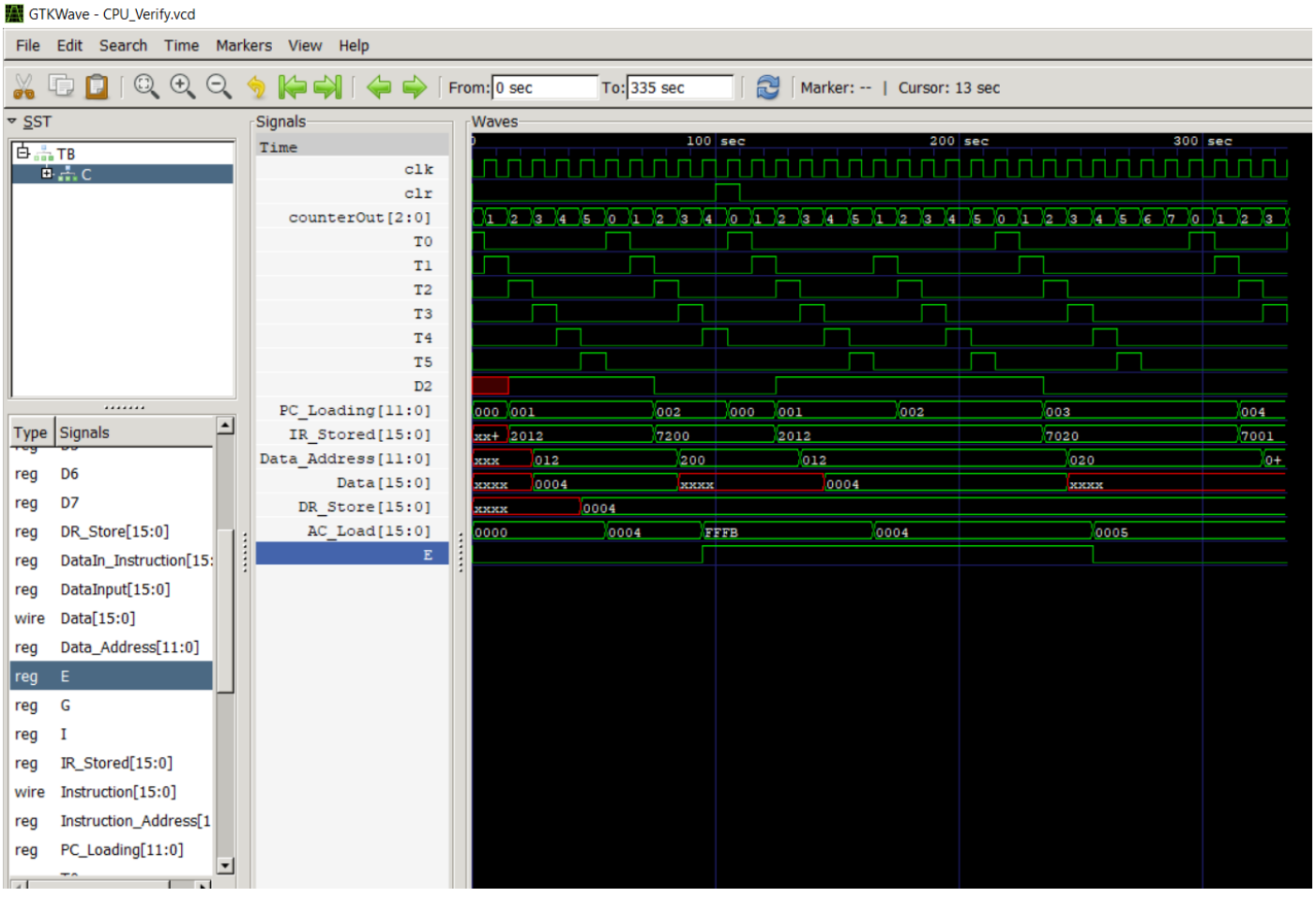
```
//Data Initialized.....................
always @(posedge clk or negedge clk) begin
memory_Data[18]=16'h0004; end
endmodule
//
//////////////////////////////////////////////////TestBench/////////////////////////////////////////////////////////
module TB ();
reg clk,clr;
wire [15:0] AC_Load;
//
initial begin
$dumpfile("CPU_Verify.vcd");
$dumpvars;
clk=0;
clr=0;
#100 clr=1;
#10  clr=0;
#400 $finish;
end
//Clock.....................
always
begin
#5 clk =~clk;
end
CPU C       (clk,clr,
            AC_Load
            );
Endmodule
```

1. Simulation's Screenshots:

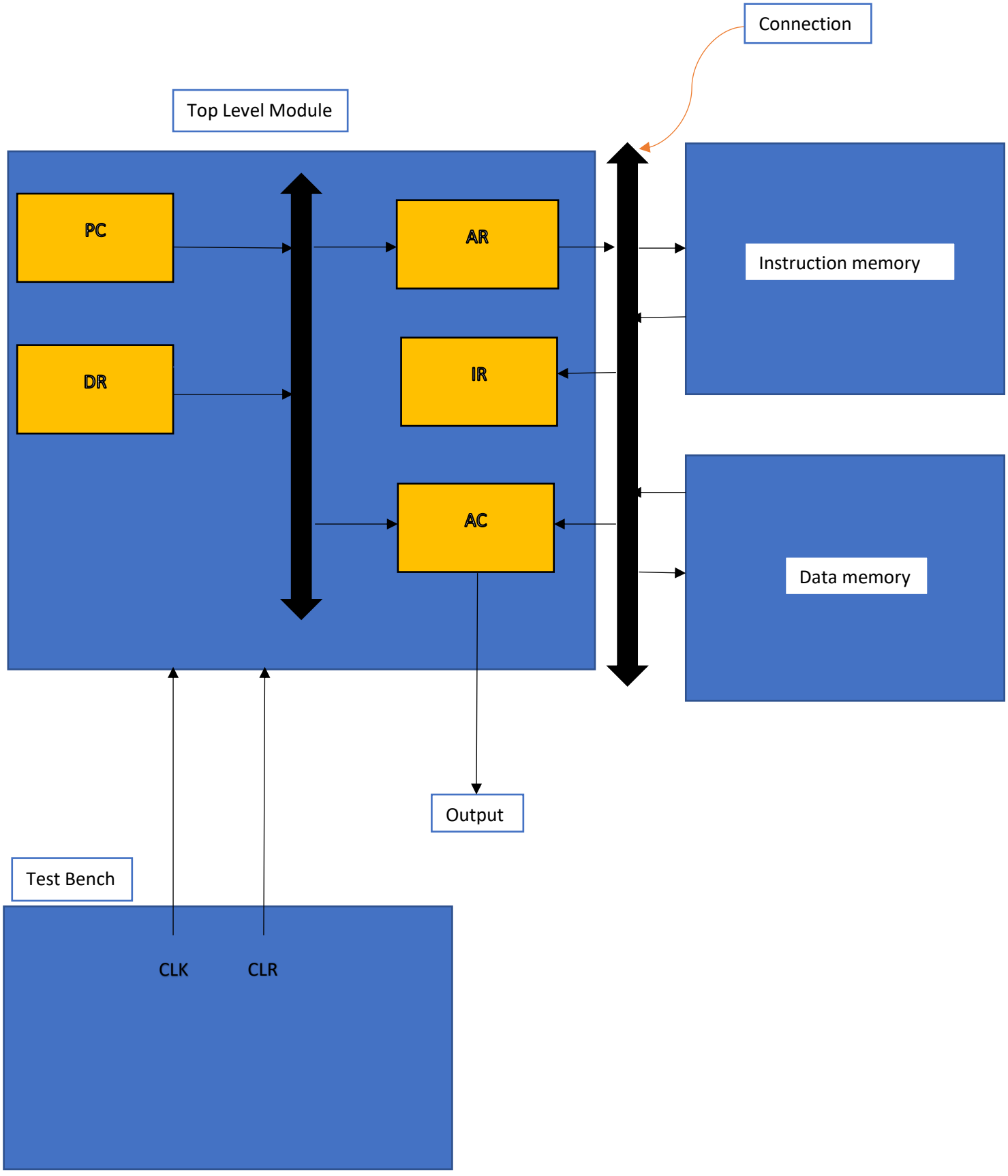
1.1. Simulation With Reset



1.2. Simulation without Reset:



2. Schematic:


Connection
Top Level Module
PC
DR
AR
IR
AC
Instruction memory
Data memory
Output
Test Bench
CLK
CLR

- Implemented Program in Test Bench:

2's complement of number	
Address	Instruction
0	2012
1	7200
2	7020
3	7001

Address	Data
12	0004